Sampling Deformed, Intersecting Surfaces with Quadtrees

Brian P. Von Herzen

Computer Science Department
California Institute of Technology

5179:TR:85

SAMPLING DEFORMED, INTERSECTING SURFACES WITH QUADTREES

Brian P. Von Herzen
California Institute of Technology
Pasadena, CA. 91125

Caltech CS Technical Report 5179:TR:85

Abstract

A quadtree algorithm is developed to render deformed, intersecting parametric surfaces with inside-outside functions. The task of adaptively sampling a surface may be broken into two parts: a subdivision mechanism for recursively subdividing a surface, and a set of subdivision criteria for determining where to subdivide. A *surface quadtree* is a collection of parametric samples arranged in a quadtree. A *restricted quadtree* is a quadtree whose neighboring elements must be the same size within a factor of two. Restricted surface quadtrees are shown to be an effective recursive subdivision mechanism. The quadtree samples are concentrated along silhouette and intersection boundaries, and in regions of high curvature, using several subdivision criteria. The recursive subdivision algorithm that finds the sample points is proven to have a complexity of $O(n)$ along boundary curves, where n is the linear resolution of the boundary in parameter space. A new *proximity subdivision criterion* concentrates samples where two surfaces potentially intersect. An extended modeling hierarchy that includes deformations is demonstrated with several examples. The initial implementation using surface quadtrees is moderately more efficient and substantially more robust than uniform sampling techniques; surface quadtrees are potentially much more efficient and robust than uniform sampling at rendering deformed, intersecting surfaces.

KEYWORDS: computer graphics, computational geometric modeling, adaptive sampling, parametric surfaces.

Contents

List of Figures ____ iii

Acknowledgements ____ iv

1 Motivation for Studying Surface Sampling Techniques ____ 1
- Introduction . . . 1
- Uniform Sampling . . . 2
- Historical Background . . . 4

2 A Recursive Subdivision Mechanism ____ 5
- Basic Approach . . . 6
- Data Structure for Quadtrees . . . 6
- Quadrilaterals vs. Triangles . . . 7
- Restricted Quadtrees . . . 8
- Neighbor-Finding Algorithm . . . 10
- Depth-first vs. Breadth-first Quadtree Creation . . . 10
- Parameter Space Wrap-Around . . . 11
- Chopping Triangles . . . 11
- Algorithm for Rendering Parametrics with Surface Quadtrees . . . 13

3 Recursive Subdivision Criteria ____ 14
- Uniform Subdivision . . . 15
- Curvature Subdivision . . . 15
- Boundary Subdivision . . . 17
- Silhouette Boundaries . . . 18
- Surface Intersection Boundaries . . . 18
- Surface Intersection Biasing . . . 19
- Proximity to other Surfaces . . . 19
- Efficient Combination of Subdivision Criteria . . . 20

4 Imaging Results ____ 22
- Fork . . . 22
- Puzzle . . . 22
- Bicycle Chainwheel . . . 24
- Nut and Bolt . . . 24
- Preliminary Silhouette Convergence Results . . . 25

5 Conclusions ____ 28
- Summary . . . 28
- Future Work . . . 28

Appendix A: Surface Quadtree Complexity Theory ____ 30

Appendix B: Glossary ____ 36

References ____ 39

List of Figures

Figure 1.A: A uniformly sampled sphere. . . . 2
Figure 1.B: Parameter space of the uniformly sampled sphere. . . . 2
Figure 1.C: A quadtree sphere. . . . 2
Figure 1.D: Parameter space of the quadtree sphere. . . . 2
Figure 2.A: A uniformly sampled chain. . . . 3
Figure 2.B: A quadtree chain. . . . 3
Figure 3: Surface Quadtree subdivision process. . . . 7
Figure 4: Cracks in a surface quadtree. . . . 8
Figure 5: Transforming a restricted quadtree into a triangular subdivision. . . . 9
Figure 6: Quadtree traversal to find **B**, the west neighbor of **A**. . . . 10
Figure 7.A: Uniform sampling of a puzzle piece. . . . 12
Figure 7.B: Quadtree sampling of a puzzle piece. . . . 12
Figure 7.C: Triangulated quadtree of the puzzle piece. . . . 12
Figure 8.A: The tangent vector subdivision criterion. . . . 16
Figure 8.B: The region obtained by connecting the four transformed vertices. . . . 16
Figure 8.C: Drain without tangent curvature subdivision. . . . 16
Figure 8.D: Drain with tangent curvature subdivision. . . . 16
Figure 9: Geometry for the boundary subdivision criterion. . . . 17
Figure 10: A cornucopia mold showing intersections of deformed parametric surfaces. 18
Figure 11: A pierced block. . . . 20
Figure 12: A fork made out of deformations of intersecting parametric surfaces. . . . 22
Figure 13.A: A puzzle piece from a six-piece puzzle. . . . 23
Figure 13.B: Three puzzle pieces assembled together. . . . 23
Figure 13.C: Completed puzzle. . . . 23
Figure 14: A bicycle chainwheel consisting of 70 primitives in boolean combination. . 24
Figure 15: Nut and bolt, rendered from deformed, intersecting superquadrics. . . . 24
Figure 16.A: Uniformly sampled chocolate chip. . . . 25
Figure 16.B: Adaptively sampled chocolate chip. . . . 25
Figure 17: Breadth-first subdivision of the parameter space of a chocolate chip. . . . 26
Figure 18: CPU time and number of squares vs. silhouette boundary resolution. . . . 27
Figure A.1: Unrestricted quadtree approximating a curve. . . . 30
Figure A.2: The area of a region around a curve. . . . 31
Figure A.3: Construction of a region surrounding curve C. . . . 31
Figure A.4: Intersection region I of radius R around the curve. . . . 32
Figure A.5: Geometric progression of a restricted quadtree. . . . 33
Figure A.6: Restricted quadtree approximating the same curve as in Figure A.1. . . . 34
Figure A.7: The measured number of squares in a restricted quadtree. . . . 35

Acknowledgements

I would like to thank my advisor, Alan H. Barr, for the basic concept of using recursive sampling as a rendering technique for deformed, intersecting parametric surfaces. Jon Leech, Wen-King Su, and Mike Newton provide programs and advice in formatting and illustrating the text. This work has been supported by the Hertz Foundation, IBM, and the System Development Foundation.

Chapter 1: Motivation for Studying Surface Sampling Techniques

Introduction

An interesting mathematical formulation for deformed, intersecting surfaces has recently been developed ([BARR84]). Robust algorithms are required that make images of these surfaces quickly and efficiently. The algorithms must be able to correctly shade and outline highly curved regions on deformed surfaces. In addition, they must accurately render such critical areas as silhouette boundaries and surface intersections.

A simple approach is to cover surfaces with a parametrically uniform grid of samples, dividing the surface into small polygons that are easy to render. The size of the polygons is determined by the highest curvature of the surface for high-quality images. Unfortunately, regions of low curvature are oversampled, wasting polygons.

An alternative approach is to adaptively sample a surface, concentrating samples where they are needed the most. A quadtree subdivision technique is presented here that concentrates samples in regions of high curvature and in other critical regions. This recursive subdivision technique is potentially much more more efficient than uniform sampling of parametric surfaces. A first implementation of the technique is moderately more efficient in computation time and storage than a uniform subdivision technique, and is much more robust than uniform sampling.

The problem of how to adaptively sample a surface is broken into two chief problems: the subdivision mechanism, which recursively samples a surface; and the subdivision criteria, which determine where subdivision should take place. A quadtree mechanism is described for recursive subdivision. Other recursive subdivision approaches exist, but the quadtree technique is especially simple. Several basic subdivision criteria are found to be effective in producing high-quality images (see Figures 1.C, 1.D, 2.B, and Chapter 4).

A parametric surface may be decomposed into a set of polygons or simple curved patches. Frequently, however, the decomposition process introduces undesirable sampling artifacts, particularly when the surface curves rapidly as a function of the parameters. Mathematical models assist in detecting and correcting these sampling artifacts.

Planar patches are explored here, under the hypothesis that much can be learned about effective sampling techniques without using higher-order patches. The basic recursive sampling technique may extend to curved patches.

A glossary of terms has been provided in Appendix B as a reference for the terminology used in this paper.

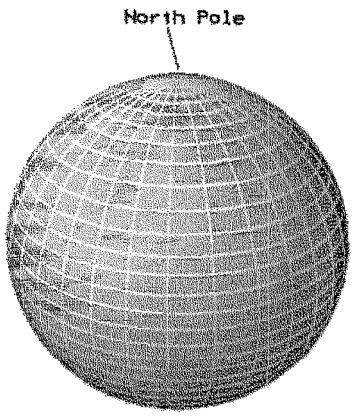


Figure 1.A: Sphere with uniform sampling in parameter space.

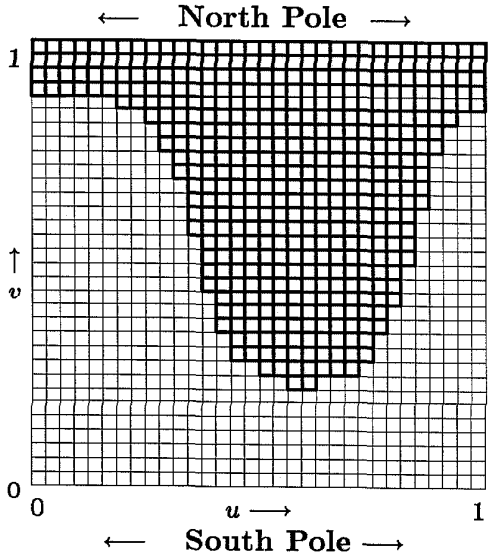


Figure 1.B: Parameter space of the sphere. Bold squares are front-facing.

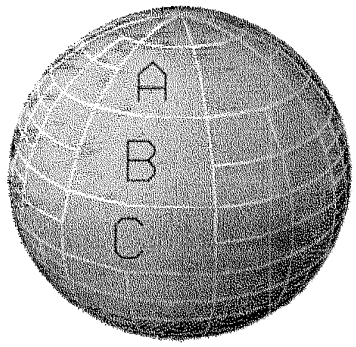


Figure 1.C: Sphere with quadtree sampling.

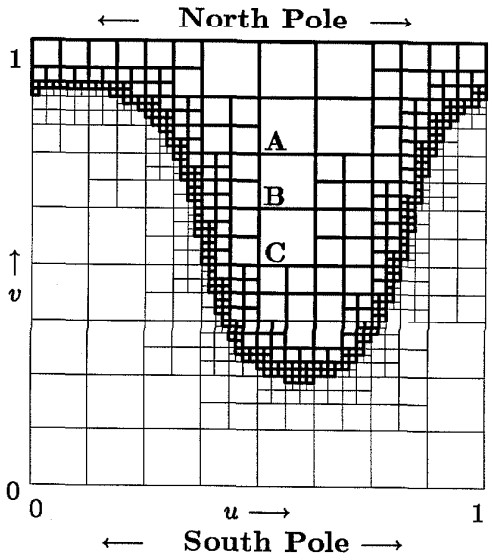


Figure 1.D: Parameter space showing quadtree subdivision of the sphere. Note the finer subdivision along the silhouette boundary.

Uniform Sampling

The simplest technique to sample parametric surfaces, such as the sphere in Figure 1, is to evenly distribute the samples on a uniform parametric grid. Although this technique is simple, it has several problems. Uniform sampling in parameter space often leads to non-uniform distributions of samples in modeling space. In Figure 1.A, the parameter lines on a sphere are dense at the north pole, and more sparse at the equator. Figure 1.B shows the parameter space of the sphere, with its polygonal decomposition. The bold squares represent visible polygons, and the fine squares are invisible polygons.

The number of uniform sample points required for high-quality images becomes prohibitively large for some surfaces, such as the deformation functions of parametric primitives ([BARR84]). Figure 2.A, Uniform Chain shows an image that took 53 minutes to compute using uniform subdivision on an IBM 4341, because most of the time was spent on minute sections of the image. The nearest link is not sampled adequately, as compared with the quadtree image of the same object, which took 45 minutes to compute (Figure 2.B: Quadtree Chain).

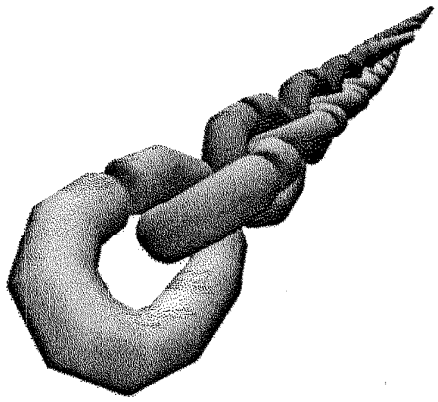


Figure 2.A: Uniform Chain that took 53 minutes to compute.

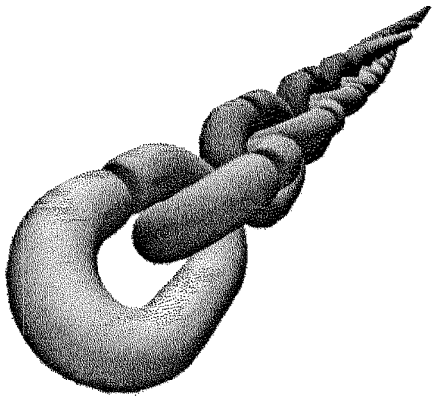


Figure 2.B: Quadtree Chain that took 45 minutes to compute.

Historical Background

Several researchers have described quadtree subdivision methods that subdivide in the plane of the image, rather than the parameter space of the surface.

Warnock describes a technique to render polygons in an image ([WARNOCK69]). The initial image is assumed to be square. The image is recursively subdivided into smaller squares until every square is entirely inside or outside the polygon, or is smaller than a pixel. All squares entirely within the polygon are colored with the polygon color.

Quadtree subdivision in screen space has been used as an antialiasing technique in ray tracing ([WHITTED80]). Rays are cast at each corner of a pixel. If the corner rays differ in intensity too much, or hit different objects, then the square pixel is subdivided into four smaller squares, and the process is repeated with each smaller square. This process continues until the desired resolution limit is reached. The average intensity of all the squares in a pixel is computed, weighted by the area of each square, to obtain the final pixel intensity.

Scan-line techniques have been used to render a variety of parametric surfaces ([BLINN78]). Hidden surface elimination is performed across each scan-line using a numerical solution technique that computes the "depth" of the surface in screen space.

Several researchers have subdivided bicubic patches to model and render geometric shapes. This sort of subdivision takes place in the parameter space of the surface, rather than in the plane of the image.

Carlson has used a parametric quadtree approach for intersecting bicubic patches ([CARLSON82]). The quadtrees are in the parameter space of the surface rather than in screen space. Two potentially intersecting patches are subdivided until the subpatches are disjoint, or are sufficiently planar to be modeled as polygons. Then the sets of intersecting polygons from the two patches are clipped to each other. The patches and polygons are stored as a model for later rendering using conventional rendering algorithms.

Schweitzer and Cobb have developed a technique for rendering bicubic patches directly instead of using polygons ([SCHWEITZER82]).

Catmull has described a technique for subdividing a bicubic surface until its subpatches are smaller than a pixel, and then shading each pixel according to the color of the subpatch ([CATMULL75]). This method avoids polygonal artifacts, but may be expensive to compute all the required samples.

Lane and Carpenter ([LANE79]) have described a recursive technique for sampling arbitrary parametric surfaces using a curvature subdivision criterion. Their curvature subdivision technique recurses until all the patches are flat to within a certain tolerance. Silhouette boundaries are more sensitive to curvature aliasing than the interior regions of an image, and usually determine the curvature tolerance.

The following chapters will discuss a new subclass of quadtrees for subdividing a surface, along with a unified set of subdivision criteria for rendering deformed, intersecting surfaces. The new type of quadtree prevents cracks from forming in a surface during subdivision. A view dependent subdivision criterion is examined that specifically subdivides at the silhouette boundary for a particular view. This reduces the amount of curvature subdivision that must be performed away from the silhouette edge of a surface. Other subdivision criteria are examined in this paper, including subdivision at surface intersections, and subdivision when two surfaces are close to each other. If any of the subdivision criteria are not met, then the region is subdivided. If the region becomes smaller than a pixel, then subdivision terminates. Regions of low curvature are represented with few samples, while regions of high curvature are represented with more samples.

Chapter 2: A Recursive Subdivision Mechanism

The problem of rendering deformed, intersecting surfaces is split into two subproblems: 1) the subdivision mechanism, determining how to subdivide a surface into simple elements; 2) the subdivision criteria, determining where additional sampling is necessary. Here we examine the recursive subdivision mechanism for adaptively sampling a set of surfaces.

A subdivision mechanism for rendering deformed, intersecting surfaces has several requirements. The mechanism should sample as finely as desired on any region of the surface. It must form a smooth transition between highly sampled and sparsely sampled regions. It should store and retrieve surface samples based on their location, so that several local sample points may be quickly obtained for further subdivision decisions. The mechanism should represent silhouette and intersection boundaries in the network of samples. Finally, the network of samples should be rendered efficiently, once they have been computed.

One solution to these requirements for a subdivision mechanism is to use a quadtree technique. The parameter space of the surface to be rendered is broken into a set of squares. Whenever more samples are needed, a square is divided into four smaller squares. Samples are obtained at the corners of each square.

Quadtrees have the property of being self-similar at all scales. This regularity is attractive for building specialized machines. Every surface element is a square, regardless of its size. Quadtrees can subdivide as much as necessary to achieve the required sampling densities. Restricted quadtrees require that neighbors be the same size to within a factor of two. This smooths the transition between finely and sparsely sampled regions. Four corner samples and a center sample are available at each square, permitting subdivision criteria to be evaluated at any square. Silhouette and intersection boundaries are represented with complexity linear in the parametric resolution (see Appendix A: Surface Quadtree Complexity Analysis).

Basic Approach

The subdivision technique uses the following steps:

---

1. The surface is sampled on a uniform parametric grid at some initial resolution.
2. Each square is evaluated using several acceptance criteria.
3. If the square isn't acceptable, then it is split into four smaller squares.
4. This process is repeated until the entire surface is sampled adequately.
5. Once the sampling is acceptable, the squares are broken into triangles.
6. The triangles are clipped at the intersection with other surfaces, forming a smooth boundary.

Figure 1.A on page 2 shows a uniformly sampled sphere. Lines are drawn on the surface of the sphere to show how it has been broken into polygons. The surface has been sampled at the corner of each square. The north pole has more samples than necessary. Figure 1.B shows the parameter space of the sphere. The parameter space consists of two parameters, u and v, which span the surface. The parameter values are normalized to the range $[0, 1]$. The top border of the parameter space corresponds to the north pole of the primitive, where $v = 1$. The parameter u ranges from 0 to 1 from left to right. The south pole is at the bottom of each net, where $v = 0$. Boldface lines in parameter space correspond to the visible parts of the surface. The light lines in parameter space correspond to the back side of the sphere.

Figure 1.C shows a sphere sampled using the recursive subdivision technique of this paper. Squares have been concentrated along the silhouette edge of the sphere, and reduced in the middle and on the back side of the sphere. Figure 1.D shows this sphere in parameter space. The sinusoidal line of small squares represents the silhouette boundary of the sphere, where many samples are needed to reduce visual artifacts of the sampling process. The north pole of this sphere has fewer samples than with uniform subdivision.

The recursive subdivision technique produces as good a picture of the sphere as the uniform subdivision technique. The uniform sphere took 109 seconds, 1365 samples, and 3354 triangles, while the quadtree sphere took 67 seconds, 785 samples, and 2254 triangles. In this example, quadtrees were more efficient than uniform grids according to all three criteria: time, number of samples, and number of triangles.

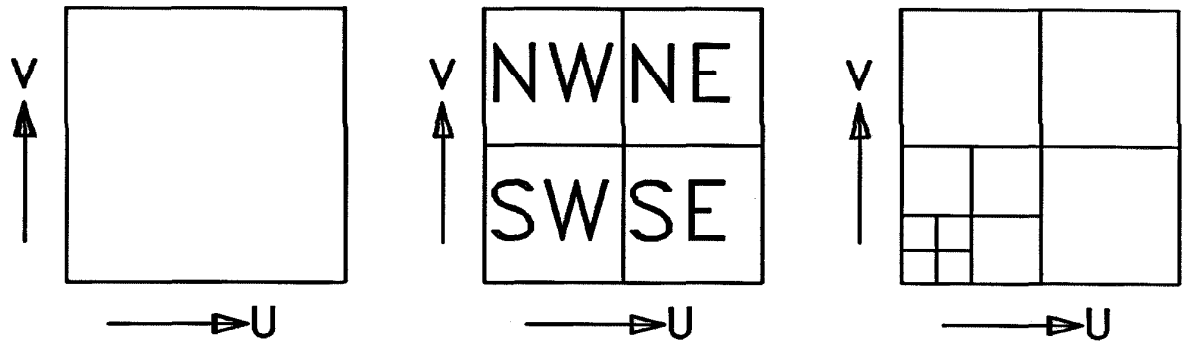


Figure 3: Surface quadtree subdivision process.

Data Structure for Quadtrees

A hierarchical tree is implemented with pointers to represent the quadtree. Samet describes basic definitions and access procedures for quadtrees ([SAMET82]). A quadtree is a tree with a branching ratio of four, in which the root represents a square, and the four children represent quadrants of the square (Figure 3: Surface quadtree subdivision process). The quadrants are labeled NW, NE, SE, SW. Each child has a pointer back to its parent, and four pointers for its children.

Several operations are supported using this data structure. Neighbor-finding algorithms exist to find neighbors of the terminal nodes of a quadtree. It is easy to add or delete elements to the quadtree. The quadtree may be stored in a compact linear form, for archival storage of the subdivision areas ([TAMMINEN84]).

There are several ways of storing the surface samples.

Each quadtree node can keep track of its four corner samples. This implies that four quadtree squares point to the same sample, resulting in an overhead of four pointers per sample.

Alternatively, the samples may be stored in a two-dimensional bucket array according to u, v parametric values. The array reduces the memory overhead for storing samples, by having only two pointers to store each sample in a list, instead of the four pointers needed for each neighboring square of a vertex.

A third alternative is to create a hash table of samples based on the u and v parametric values of each sample. This has the benefit of evenly distributing samples throughout the array, and also requires two pointers per sample.

The implementation in this paper used the two dimensional bucket array to store the samples, providing a simple way to verify the basic algorithm. Future implementations may benefit from a hashing function on fixed-point representations of the u and v parameters.

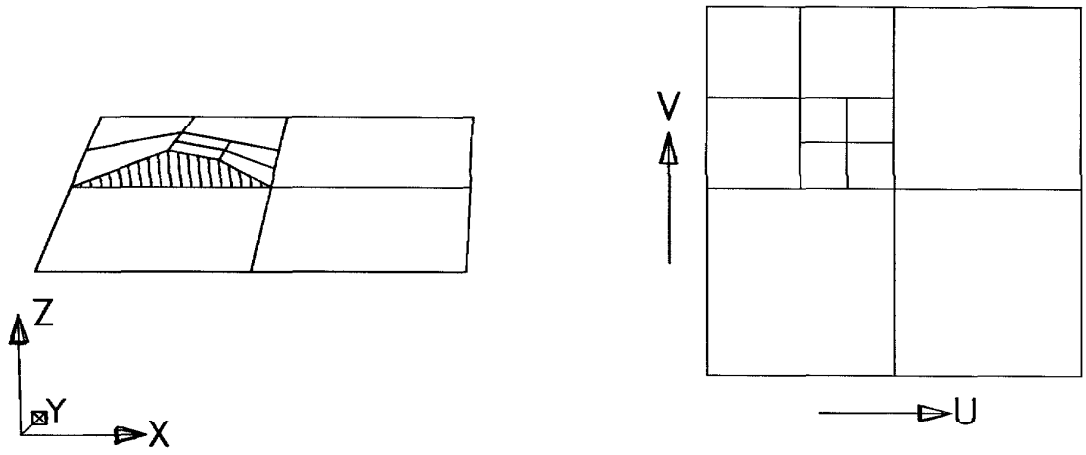


Figure 4: Cracks in a surface quadtree.

Quadrilaterals vs. Triangles

In rendering a surface represented with a quadtree, several problems exist with directly rendering the squares at the bottom of the quadtree. Cracks appear in the surface wherever small squares adjoin larger squares (see Figure 4: Cracks in a surface quadtree). The edge of a big square cannot match the edges of small squares, because they can leave the plane of the big square. To eliminate cracks, it is necessary to construct a *planar subdivision* of the parameter space ([KIRKPATRICK83]). Kirkpatrick defines a planar subdivision to be a collection of line segments whose intersections are restricted to segment endpoints. A *triangular subdivision* is a planar subdivision whose regions always have three edges. Although a quadtree is not a planar subdivision, since vertices touch edges, a triangular subdivision may be constructed from the quadtree. The approach is to take a sample at the center of each quadtree square, and link the center sample with all the samples on the edges of the square. This reduces the square to a set of non-overlapping triangles.

Restricted Quadtrees

The adaptive sampling process relies on the limited sampling information available to decide whether to obtain additional samples. If there aren't enough samples to begin with, the adaptive sampler will fail. One indication to subdivide is that the neighboring squares have subdivided for some reason. A new type of quadtree, called a *restricted quadtree*, propagates this subdivision information to neighbors.

The rule about restricted quadtrees is that neighboring squares may differ in size by at most a factor of two. This prevents sudden changes in the sampling rate over a surface. A smooth transition is produced between highly sampled regions and sparsely sampled regions, minimizing artifacts associated with the change in sampling rate.

Restricted quadtrees have several other advantages over regular quadtrees. Restricted quadtrees sample more near important features, making the algorithm more robust. They also improve the robustness of curve-finding algorithms by sampling more near boundary curves.

Restricted quadtrees are easily broken into triangles. Surface cracks are eliminated by guaranteeing that the triangulation of the surface forms a planar subdivision.

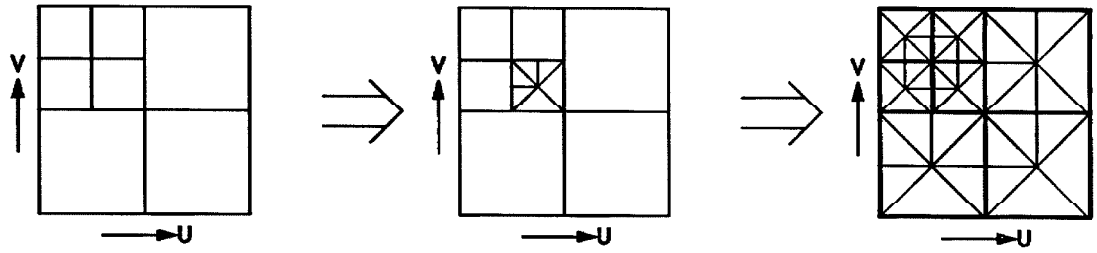


Figure 5: Transforming a restricted quadtree into a triangular subdivision.

A square in a restricted quadtree is decomposed into triangles using a simple test. For each edge of the square, the size of the square is compared with the size of the neighboring square that shares the edge (Figure 5). If the neighbor is larger, then a single triangle is formed between the edge and the center sample of the square. If the neighbor is the same size or smaller, then the edge is split in half, and two triangles are formed using the two half-edges and the center sample. A triangular subdivision is formed with this algorithm, eliminating cracks in the surface.

A quadtree square may have at most two larger neighbors, since two of the neighbors must be siblings of the square. Therefore, each square in a restricted quadtree produces six triangles if it has two larger neighbors, and eight triangles if it has zero larger neighbors. These triangles always form a triangular subdivision.

It is possible for two squares of the same size to be joined with just one triangle per edge, but this produces a preponderance of lines that run 45 degrees to the parametric directions. For many parametric surfaces, the principal directions of curvature lie along parametric directions. Polygons with most edges lying 45 degrees to the principal directions produce visual artifacts in highly curved regions.

The ideal solution would be to have polygon edges match the principal directions of curvature of the surface. Since the edges would match the direction of maximum curvature, polygonal artifacts due to curvature would be minimized. This may be a promising area for future research.

The current implementation makes the assumption that the principal directions of curvature match the parametric directions of the surfaces being rendered. This approximation is reasonable for superquadrics under moderate deformations. When the approximation is incorrect, the adaptive sampler must spend additional time sampling regions of high curvature. The approximation is implemented by forcing most of the triangle edges to lie along parametric directions. Forcing equal-sized squares to split into two triangles per edge causes more triangle edges to be aligned with the parametric directions, and usually the principal curvature directions. This is the motivation for splitting quadtree squares always in six to eight triangles, as opposed to four triangles.

Splitting quadtree squares (such as those in Figure 5) into two triangles would not form a planar subdivision. While two triangles per square are sufficient for uniform sampling, they do not provide enough flexibility for adaptive sampling.

Neighbor-Finding Algorithm

Quadtree triangulation requires that neighbors be found easily. Samet has described an efficient technique for finding neighbors, which searches for the smallest common ancestor between a square and its neighbor ([SAMET82]). The path used to travel from the common ancestor to the square is "reflected" to find the neighbor (see Figure 6). Samet has proven that a neighbor-finding algorithm takes an average of four node traversals of the quadtree, for quadtrees of arbitrary size ([SAMET82]).

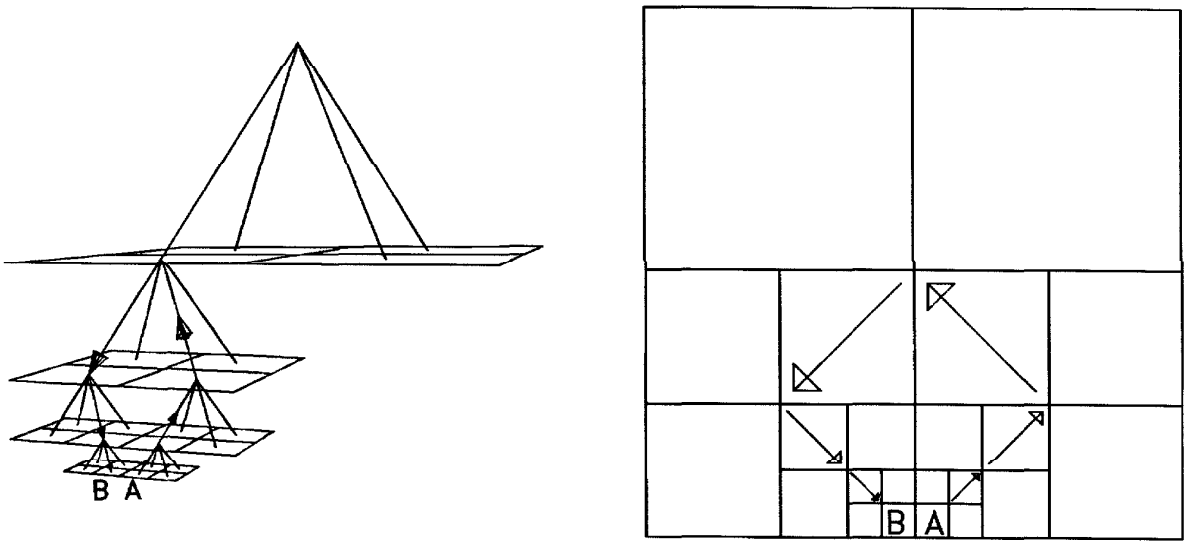


Figure 6: Quadtree traversal to find **B**, the west neighbor of **A**. Arrows represent the traversal order.

Depth-first vs. Breadth-first Quadtree Creation

A tradeoff must be resolved whether to create the surface quadtree in a depth-first or a breadth-first fashion. Depth-first search is a quicker way to traverse a tree, since each node of the tree needs to be visited only once. The entire tree need not be stored, but only one branch at a time. Unfortunately, restricted quadtrees seem difficult to create in a single depth-first traversal. Branches of the quadtree that are being created may affect branches that have already been fully developed, which appears to be incompatible with a depth-first approach.

In this paper, a breadth-first algorithm is used, since it is sufficient for implementing restricted quadtrees. Each new square is tested against the subdivision criteria with each traversal of the tree. Subdivision occurs whenever a square fails some criterion. This process continues until all the squares pass the subdivision criteria.

Breadth-first subdivision discovers features that are missed by depth-first subdivision. When the subdivision is performed breadth-first, neighbors have a chance to propagate subdivision information to each other. All of a square's neighbors are not available in a depth-first traversal, so subdivision information cannot always be propagated. In addition, interactive applications benefit from a breadth-first approach where one can view successively improved versions of the entire image.

Tree traversal order is an important issue concerning the efficiency of a quadtree implementation.

Further work will have to clarify the advantages of the various traversal schemes.

Parameter Space Wrap-Around

A parametric surface can be thought of as a transformation of the unit square into another shape. Some parametric surfaces are periodic: their edges wrap around to meet each other. A seam may be created where the boundaries of the unit square join. For closed surfaces it is important that the sample points match at edges of the unit square in parameter space. The boundaries $u = 0$ and $u = 1$ must contain the same sample points. For tori, the sample points on the boundaries $v = 0$ and $v = 1$ are also constrained to be identical.

Restricted quadtrees allow an interesting implementation of this constraint. We define squares on the west edge of the quadtree to be neighbors of squares on the east edge of the quadtree. For supertori, the squares on the north edge of the quadtree are neighbors with the squares on the south edge of the quadtree. Then the techniques for avoiding cracks between neighbors apply, and the seam at the parametric boundary is eliminated.

When two surfaces intersect, a similar problem arises in joining the boundaries of two surfaces. A polygonal representation of one surface boundary will not necessarily match the representation of the other boundary. Cracks appear between the surfaces wherever the sample points are not coincident. The ideal solution would be to force the surface intersection boundaries of the two surfaces to contain the same sample points. Unfortunately, it is difficult for quadtrees of two surfaces to contain exactly the same boundary sample points, since the two quadtrees are independent. The problem is explored in more detail in Chapter 3.

Chopping Triangles

After the squares of the quadtree are broken into triangles, the triangles are tested against inside-outside functions of other surfaces, and are clipped at the intersection boundary. This removes the ragged appearance that results when all the border triangles are drawn (Figure 7.A: Uniform sampling of a puzzle piece without triangle chopping). Figure 7.B shows the effects of chopping boundary triangles and quadtree sampling. This technique dramatically improves the quality of the images. Figure 7.C shows the triangulated quadtree of the parameter space, with triangle chopping.

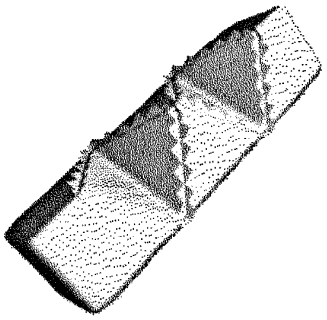


Figure 7.A: Uniform sampling of a puzzle piece, without triangle chopping.

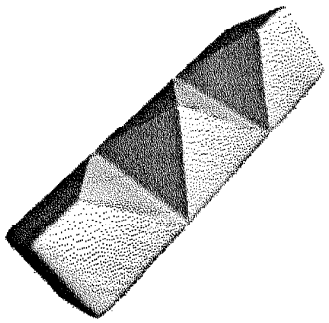


Figure 7.B: Quadtree sampling of a puzzle piece, with triangle chopping.

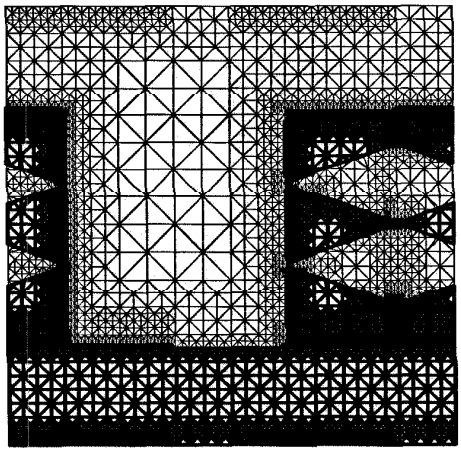


Figure 7.C: Triangulated quadtree of the puzzle piece.

Algorithm for Rendering Parametric Surfaces with Surface Quadtrees

To summarize the subdivision mechanism for parametric surfaces:

---

1. Uniformly subdivide the unit square down to an initial sampling level sufficient to crudely resolve the object.
2. For each square in the quadtree, test the square against the subdivision criteria.
3. Split the square into four smaller squares if it doesn't pass, and split all neighboring squares that are larger than the test square, including wrap-around neighbors.
4. Repeat until all squares pass the subdivision criteria, or the subdivision limit is reached.
5. Split all squares into suitable triangles for rendering.
6. Chop the triangles with the inside-outside function for the surface.

Chapter 3: Recursive Subdivision Criteria

A set of recursive subdivision criteria are needed to determine where subdivision should occur. The criteria should include a method to detect surface curvature, and to locate silhouette and surface intersection boundaries. When two surfaces potentially intersect, subdivision should resolve the interference question.

Highly deformed regions must be sampled finely to avoid artifacts that are readily visible. The visible curvature of silhouette and other surface intersection boundaries should guide subdivision in these regions. The proximity of two surfaces should control subdivision for potential interference.

The philosophy of the method is to mathematically measure the visible "badness" of each part of the surface, and subdivide until the "badness" of each patch is less than a prescribed tolerance. The goal is to diminish the effects of functional and parametric aliasing artifacts; the artifacts generally occur where the gradient or derivatives of the function's visible aspects have the largest magnitude.

Several coordinate systems are useful in describing various subdivision criteria. Parameter space, denoted by u,v coordinates, is a two-dimensional space frequently limited to the unit square, spanning the domain of the parametric surface. The parametric surface is embedded in a three-dimensional space called the modeling space. The parametric surface equation $f(u,v) = \begin{pmatrix} x \\ y \\ z \end{pmatrix}$ maps a point (u,v) in parameter space onto a point $\begin{pmatrix} x \\ y \\ z \end{pmatrix}$ in modeling space. Screen space uses the perspective viewing transformation $\mathbf{M}$ to map modeling space coordinates $\begin{pmatrix} x \\ y \\ z \end{pmatrix}$ onto a three dimensional screen-centered coordinate system $\begin{pmatrix} x_s \\ y_s \\ z_s \end{pmatrix}$.

$$\mathbf{M} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} x_s \\ y_s \\ z_s \end{pmatrix}.$$

Screen space is useful for determining the visual size of a feature, and for rendering the final image.

It is attractive to use screen space for many of the subdivision criteria, because subdivision should halt if a feature is not visible. This may occur if the feature is too far away, or occluded.

The following set of subdivision criteria provide robust set of samples for a polygon rendering system. These samples could provide an initial guess for ray-intersections in a ray-tracing system that renders deformed, intersecting objects. Many numerical iteration techniques benefit from an accurate first guess at the solution.

Uniform Subdivision

It is easy to imagine starting the recursive sampling process with an initial $n \times m$ grid of samples. Unfortunately, this may require creating a quadtree at each square in the $n \times m$ grid. It is much easier to have one quadtree per parametric surface, and sample initially using a $2^n \times 2^n$ grid. A simple criterion measures the size of a square in parameter space, and forces subdivision until the size is less than a threshold value.

This subdivision criterion is useful for establishing an initial sampling grid for other subdivision criteria. The sampling criteria need some limited information about the surface to make initial decisions about subdivision; the uniform subdivision criterion provides this information.

Curvature Subdivision

Curvature subdivision uses modeling space and screen space measurements to control subdivision. The essential task is to obtain some measure of the local curvature of an object. Where the curvature is high, a region is subdivided. The subdivision process terminates when a region becomes sufficiently planar, or when the region becomes smaller than a pixel. Curvature estimation may be performed in several ways. Lane and Carpenter ([LANE79]) describe a curvature estimation technique that measures the distance from a surface to its planar approximation. The distance indicates the error introduced by the approximation.

In contrast to the Lane and Carpenter algorithm, the curvature subdivision criterion used here computes normal vectors from the normal vector equation for the surface. Normal vectors are available at the corners of each quadtree square for shading computations; and a simple vector equation of these normals provides a curvature subdivision criterion. Every adjacent pair of normal vectors $(\underline{N_1}, \underline{N_2})$ of a square must satisfy

$$(1 - \underline{N_1} \cdot \underline{N_2}) < \epsilon.$$

The square is tested only if larger than a pixel. This curvature test indicates the angle between the normal vectors across the square. Since the vectors N_1 and N_2 are normalized, the left-hand side of the above equation may be rewritten $(1 - \cos\theta)$, where θ is the angle between N_1 and N_2. The actual curvature would be given by $d\theta/dx$, where x is the spatial coordinate in the direction of the line joining the two normal vectors. The dot product equation is sufficient for obtaining an indication of the curvature of a surface, without computing the actual curvature.

Tangent vectors are also used in the same curvature test. They are computed using the corners of each square to obtain finite approximations to tangent vectors in the u and v parametric directions. The tangent vectors are normalized to unit length. Adjacent tangent vectors must comply to the same dot product equation as the normal vectors, subject to the image size constraint. For every pair of adjacent tangent vectors $(\underline{T_1}, \underline{T_2})$,

$$(1 - \underline{T_1} \cdot \underline{T_2}) < \epsilon.$$

The tangent vector subdivision criterion is found to be necessary because it is possible to have a highly curved patch whose normal vectors all point in the same direction. Distorting a rectangle into a U-shape is a good example (Figure 8.A, 8.B). The sheet stays in the plane, but its tangent vectors curve from one side of the rectangle to the other. It is important that such highly curved regions be sampled finely. When a surface is bent sharply, there is a good chance that aliasing

will occur (Figure 8.B) unless the spatial sampling frequency is increased (Figure 8.A). The area that should be shaded differs from what is obtained by simply connecting rectangle vertices. Fine sampling is required for proper shading in such circumstances. The effect of U-shaped polygons is shown in Figure 8.C: Drain without tangent curvature subdivision. Tangent curvature subdivision eliminates the problem (Figure 8.D: Drain with tangent curvature subdivision), improving the robustness of the curvature subdivision criterion.

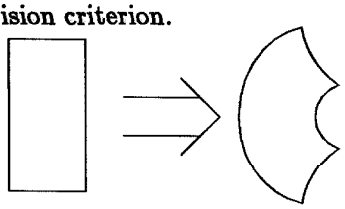


Figure 8.A: The tangent vector subdivision criterion samples U-shaped polygons accurately.

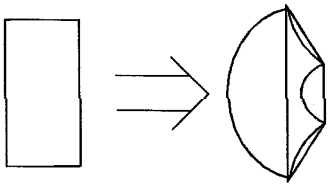


Figure 8.B: The region obtained by connecting the four transformed vertices of the rectangle does not represent the U-shaped polygon well.

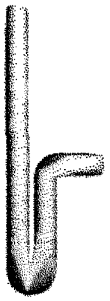


Figure 8.C: Drain without tangent curvature subdivision. Note the shading discontinuities at the corners.

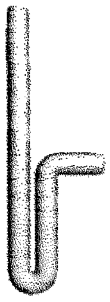


Figure 8.D: Drain with tangent curvature subdivision. The shading function is much smoother at the corners.

Boundary Subdivision

Silhouette and other surface intersection boundaries can be represented as curves in parameter space. They are susceptible to aliasing if not adequately sampled. Boundaries should be represented accurately, as they curve in screen space. One approach is to subdivide squares crossing boundaries until they are smaller than a pixel in screen space. This technique is robust, but expensive, in computing the boundary of the surface.

An alternative technique is explored here, which attempts to measure the linearity of the boundary in screen space. Subdivision occurs until either the boundary is linear in screen space, or the square is the size of a pixel. If the boundary is linear, then triangle chopping produces an accurate boundary.

Finding silhouettes and intersection boundaries may be thought of as a problem of accurately finding constant-value contours of a bivariate function. The contours are defined by functions that are zero at the boundary, positive on one side of it, and negative on the other.

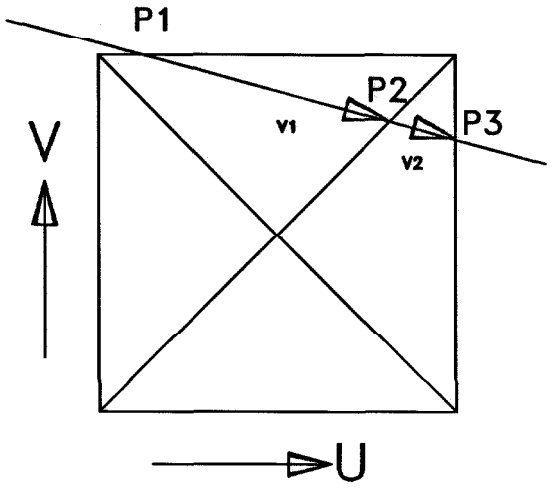


Figure 9: Geometry in parameter space for the boundary subdivision criterion. Note that any straight line intersecting the square must cross the edges or diagonals of the square in at least three places, $\underline{P_1}$, $\underline{P_2}$, and $\underline{P_3}$.

The test for linearity of a boundary curve across a quadtree square is computed using approximate tangent vectors of the boundary curve. Figure 9 shows a square with four corner vertices connected to a center vertex. Note that any straight line crossing the square must intersect the lines of the square in at least three places, $\underline{P_1}$, $\underline{P_2}$, and $\underline{P_3}$. These intersection points are found using a variation of *regula falsi* iteration in screen space (see glossary). The three intersection points in screen space are used to obtain two vectors, $\underline{V_1}$ and $\underline{V_2}$, which are approximate tangent vectors of the boundary curve.

$$\underline{V_1} = \underline{P_2} - \underline{P_1}$$

$$\underline{V_2} = \underline{P_3} - \underline{P_2}$$

The two boundary vectors are normalized, and the cross-product is taken, to obtain the sine of the angle between them. This indicates the boundary curvature K across the square.

$$K = \left| \frac{\underline{V_1}}{|\underline{V_1}|} \times \frac{\underline{V_2}}{|\underline{V_2}|} \right|$$

K^2 may be computed instead of K, eliminating the need to compute square roots. If K^2 exceeds a threshold, then the square is subdivided, unless the square is smaller than a pixel in screen space. The triangle chopping technique smooths the edges created.

Silhouette Boundaries

The eye is quick to pick up slight irregularities at the sharp border of an object, which has high spatial frequencies. Curvature aliasing is easier to see near the silhouette boundary of a surface than on the interior, requiring additional subdivision on the silhouette.

The bivariate function for silhouette subdivision is the dot product between the surface normal $\underline{N}$ and the view vector $\underline{E}$. When $(\underline{N} \cdot \underline{E}) = 0$, the sample is on a silhouette boundary. When $(\underline{N} \cdot \underline{E}) < 0$, the sample is facing towards the viewpoint. And when $(\underline{N} \cdot \underline{E}) > 0$, the sample is facing away from the viewpoint.

Recursion stops when the curvature of the silhouette boundary in screen space is less than a threshold value, or when the square is smaller than a pixel. The second termination condition prevents sharp corners from causing infinite recursion.

Surface Intersection Boundaries

Boolean subdivision is similar to silhouette subdivision, except that the inside-outside function of the intersecting surface is used as the bivariate contouring function. A boolean operator returns a positive value when outside the neighbor and a negative value when inside the neighbor. (See [BARR83] for inside-outside functions of superquadrics). Figure 10: Cornucopia shows an example of boolean subdivision.

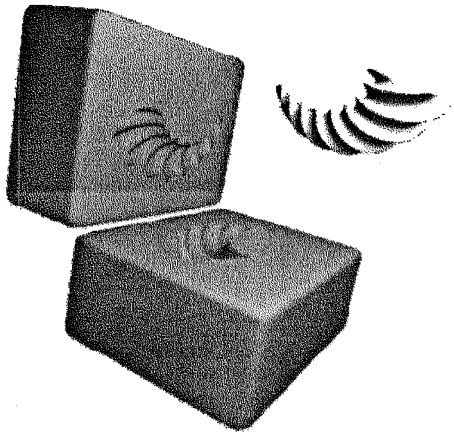


Figure 10: A cornucopia mold showing intersections of deformed parametric surfaces.

Boolean subtraction is implemented in the following manner. Suppose that we want to subtract one surface from another. Then we keep all samples on the first surface that are inside of the second; and we keep all samples on the second surface that are outside the first. Inside and outside are defined

by implicit inside-outside functions for the parametric surfaces. Squares on the surface intersection boundary are split, until the boundary curvature is sufficiently low. For boolean subtraction, the normal vectors of the subtracted surface are negated, so that shading and backface culling will be done correctly.

Surface Intersection Biasing

One problem is that the inside-outside function of a surface may not correspond to the parametric function of the surface exactly, because of the finite precision arithmetic being used. This may cause the two surfaces to overlap slightly, resulting in a fringe where the two surfaces touch. The ideal approach would be to connect the surface quadtrees of two intersecting surfaces together, so that intersection boundaries match exactly. Unfortunately, this has been a difficult problem for the current implementation, since each surface quadtree is processed independently.

The current implementation biases the inside-outside function slightly to prevent the two sets of polygons from touching each other. The bias value is set to be greater than the expected uncertainty in the inside-outside function of the surface. This is not an ideal solution; sometimes a visible crack is produced between the two surfaces, as in Figure 7.B: Puzzle piece.

Proximity to other Surfaces

Proximity subdivision is a modeling space criterion, which explores parametric surface regions that are near to each other. It is important to determine if these surfaces intersect. Additional samples are obtained in the region where the inside-outside function approaches zero. A surface is subdivided until either a zero-crossing is found or an extremum is found to within a certain percentage. This subdivision criterion helps to resolve potentially intersecting surfaces. Figure 11: Pierced Block shows the effectiveness of proximity subdivision in finding intersections. A small needle was subtracted from a superquadric block. The intersection would have gone unnoticed were it not for the proximity subdivision criterion.

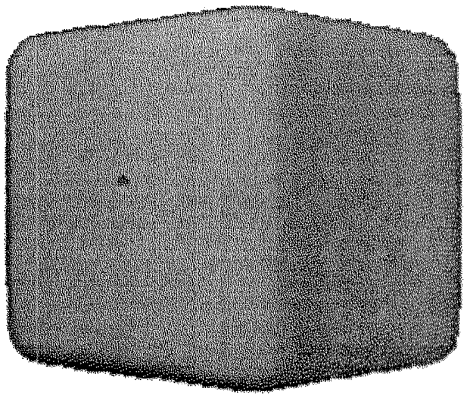


Figure 11: Pierced block. A test of the proximity subdivision criterion.
A small needle pierces the superquadric, which would have gone undetected without proximity subdivision.

Efficient Combination of Subdivision Criteria

A quadtree square must subdivide if it fails any of the subdivision criteria. For instance, one may wish to perform the easiest tests first, postponing criteria that are expensive to compute. Once the square passes all subdivision criteria (or is culled from further consideration), a flag is set to indicate that the square should not be reexamined during the next breadth-first passes of the quadtree.

A simple technique can determine an approximate ordering for the various subdivision criteria, if they are independent of each other. Given a set of n tests that may be performed in arbitrary order, we want to find the ordering that maximizes the probability, as a function of time, of obtaining a decision to subdivide. Let t_i be the time to compute test i, and p_i be the probability of the test resulting in a decision. Each test has a decision rate, $r_i = p_i/t_i$, which has units of decision probability per unit time. To maximize the decision probability, we choose the test with the maximum decision rate first, performing other tests later if a decision has not yet been reached. This basic approach guides the subdivision criterion ordering used here.

Two of the tests in the algorithm halt the subdivision process: the backface culling test, and the surface intersection culling test. It is important to evaluate some of the subdivision criteria before the culling tests, because the criteria may rely on information that would be culled. The proximity subdivision test is an example: it must examine squares that would normally be culled by the surface intersection culling test. The proximity test must be performed before the culling test, so that potentially intersecting regions are found.

The following ordering of tests has generated the images in this paper:

```
If the square is bigger than the initial sampling grid, then subdivide.
Else if the square is facing away from the viewer, then stop subdividing.
Else if the proximity test shows that the square potentially intersects
   another surface, then subdivide.
Else if the square is culled by a surface intersection, then stop subdividing.
Else if the square fails the flatness test, then subdivide.
Else if the square fails the intersection boundary curve straightness test,
   then subdivide.
Else if the square fails the silhouette boundary curve straightness test,
   then subdivide.
Else stop subdividing the square.
```

Chapter 4: Imaging Results

The following illustrations show the variety of deformed, intersecting parametric surfaces that may be rendered using the adaptive sampling techniques described.

Fork

The fork is modeled using a one-dimensional tapering function to profile the width of the fork. Another one-dimensional tapering deformation adjusts the thickness of the fork along its length. The tines of the fork are created by subtracting three discs centered at the end of the fork. The fork is bent in several places to create the desired profile.

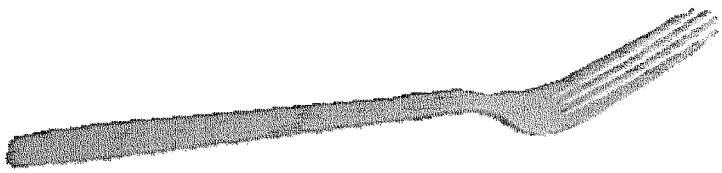


Figure 12: A fork made out of deformations of intersecting parametric surfaces.

Puzzle

The puzzle is modeled with six identical pieces that fill the volume of the interior of the puzzle. An individual piece is formed by taking a superquadric block, and subtracting two similar blocks to cut wedges in the piece (Figure 13.A: Puzzle Piece). Three pieces may be assembled to form the cluster shown in Figure 13.B: Triple Cluster. The completed puzzle is formed with a left-handed triple cluster and a right-handed triple cluster (Figure 13.C: Hex Puzzle).

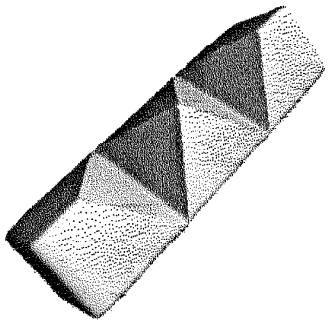


Figure 13.A: A puzzle piece from a six-piece puzzle.

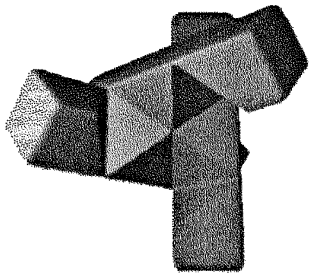


Figure 13.B: Three puzzle pieces assembled together.

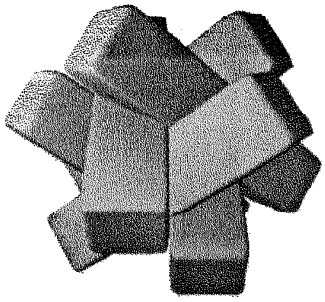


Figure 13.C: Completed puzzle.

Bicycle Chainwheel

The chainwheel uses the boolean subtraction operation extensively. Fifty-two cylinders are subtracted from a disk to form the teeth for the gear. The proximity subdivision criterion helps to locate the position of the teeth. Deformed cylinders cut holes in the gear to make it lighter. Superquadric cranks and pedals are added after the cutting process.

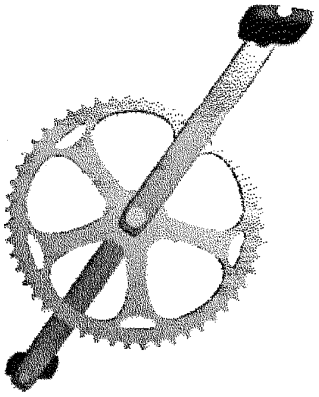


Figure 14: A bicycle chainwheel consisting of 70 primitives in boolean combination.

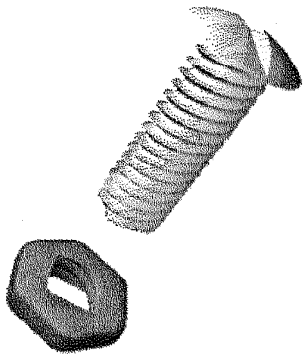


Figure 15: Nut and bolt, rendered from deformed, intersecting superquadrics.

Nut and Bolt

A type of screw thread may be formed by taking a superquadric with a square profile and twisting it for several revolutions. This thread is subtracted from the nut, and merged with the bolt head, to form the nut and bolt combination. The thread before twisting had a square cross-section, and the twist does not affect the cross-section, but only the axial profile. Where the surface of the nut intersects the thread, the profile is square. But the bolt actually threads the nut, because they both have the same twisting profile.

Preliminary Silhouette Convergence Results

A chocolate chip is modeled using tapering bending deformations (Figure 16). A series of pictures of the parameter space of the chocolate chip are shown, at successively higher resolutions. (Figure 17: Breadth-first subdivision of the parameter space of a chocolate chip) Each successive picture is rendered with a factor of 2 higher silhouette resolution.

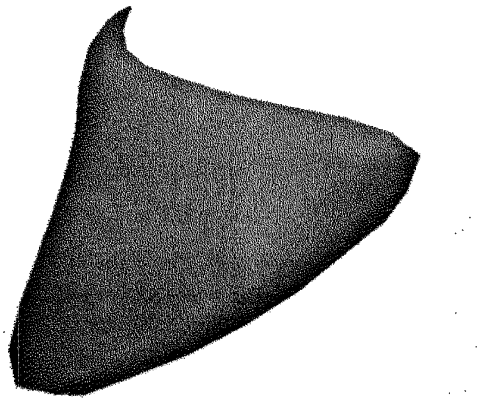


Figure 16.A: Uniformly sampled chocolate chip.

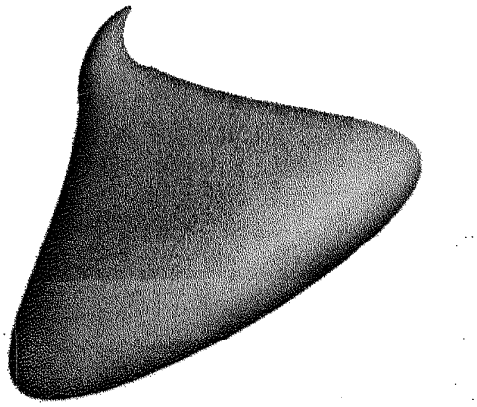
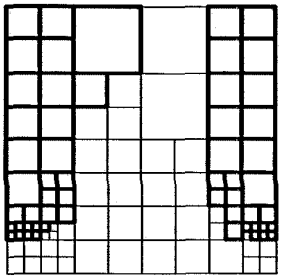
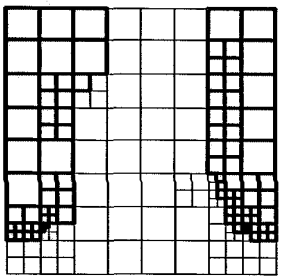


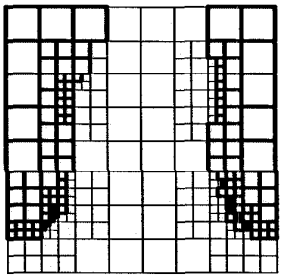
Figure 16.B Adaptively sampled chocolate chip.



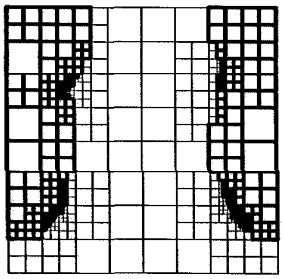
Generation 5



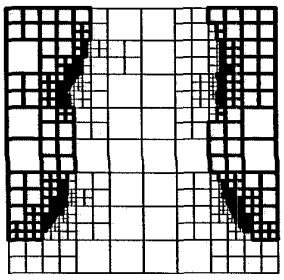
Generation 6



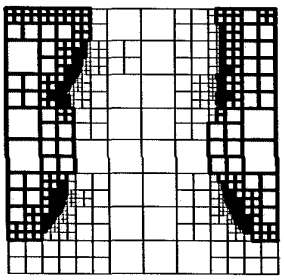
Generation 7



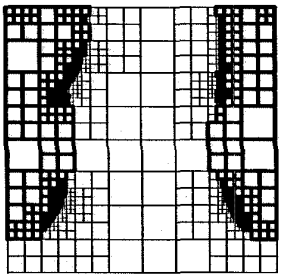
Generation 8



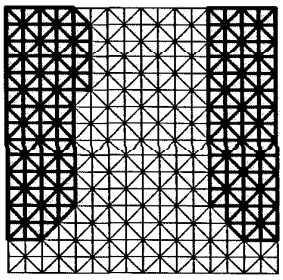
Generation 9



Generation 10



Generation 11



Uniform Sampling

Figure 17: Breadth-first subdivision of the parameter space of a chocolate chip.

Image	Silhouette Resolution (1/screen-size)	CPU Time (seconds)	Number of Squares
1	8	13	129
2	16	25	217
3	32	38	341
4	64	59	561
5	128	96	817
6	256	128	1081
7	512	158	1257
8	1024	189	1485
9	2048	237	1797

Table 1: Computational requirements for several silhouette boundary resolutions of the chocolate chip.

Figure 18 shows the same results plotted on a double logarithmic scale, illustrating the trend in computation as a function of the silhouette boundary resolution.

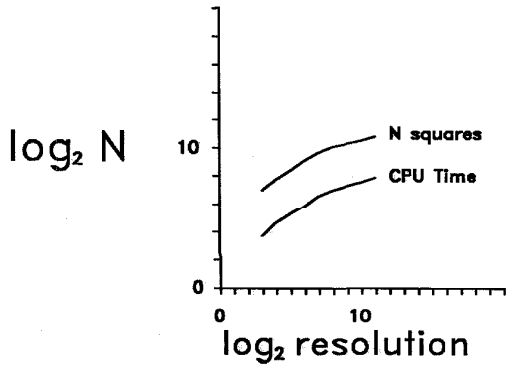


Figure 18: CPU time and number of squares vs. silhouette boundary resolution, on a log-log plot.

If subdivision were performed everywhere along the silhouette boundary down to some parametric size limit, the complexity ratio between images would be $O(n)$, where n is the silhouette resolution (see Appendix A: Surface Quadtree Complexity Analysis). The complexity of the algorithm actually used is at most $O(n)$, because subdivision occurs only where the boundary is curved in screen space. The data in Figure 18 suggests that the silhouette convergence algorithm is better than linear in computation time and storage as a function of increasing boundary resolution, for this example.

Chapter 5: Conclusions

Summary

Surface quadtrees are an effective means for rendering deformed, intersecting parametric surfaces. The adaptive sampling problem may be decomposed into two subproblems: the mechanism for subdivision, and the subdivision criteria. High-quality images of these parametric surfaces have been created using a simple subdivision mechanism, and a small set of subdivision criteria.

A modified quadtree technique, called *restricted quadtrees*, creates smooth transitions between coarsely and finely sampled regions. This makes the sampling algorithm more robust, by forcing additional sampling near important features. It is easy to generate suitable triangular subdivisions of the surface with restricted quadtrees. Many of the triangle edges are oriented parallel to the parametric directions, which frequently correspond to the principal directions of curvature for superquadrics. Triangles with a high aspect ratio in parameter space are avoided.

Curvature, silhouette, boolean, and proximity subdivision techniques provide a robust set of criteria for recursively sampling parametric surfaces. These techniques have been more efficient than uniform subdivision at producing high-quality images of deformed, intersecting surfaces.

Curves such as intersection boundaries and silhouette edges may be computed at a cost of less than $O(nL)$, where n is the linear resolution in parameter space, and L is the length of the boundary in parameter space.

Future Work

Surface quadtrees are a special class of recursive subdivision. Other sorts of recursive subdivisions may be attractive at optimizing the performance of parametric surface rendering.

For instance, surface quadtrees have suboptimal efficiency when the principal directions of the surface don't line up with parameter lines. These principal directions indicate the directions of maximum curvature and minimum curvature on the surface. If a primitive is twisted, then the optimal breakdown into polygons is twisted in parameter space. Since the quadtree can not shear in parameter space, the optimal solution is not possible. General triangular subdivision techniques may be more efficient in minimizing the number of samples required on a surface. Coordinate systems that conform to the principal directions of a surface may provide attractive ways to reduce sampling rates.

The minimum distance from a point in modeling space to a parametric surface could be used as an inside-outside function. Distance information may be used to solve the interference problem between objects, and to provide for an intelligent initial exploration of surfaces that potentially intersect. Unfortunately, it is a difficult problem to compute the distance function of an arbitrarily deformed body.

The use of non-planar surface elements may be attractive in reducing the amount of subdivision required, since the slope discontinuities of polygon edges could be eliminated. Visual artifacts, such as bright and dark bands called mach bands, appear at these slope discontinuities. Quadratic patches, such as Steiner patches, have continuous slope, and may alleviate the problem ([SEDERBERG84]). Higher-order surfaces are bound to require much additional computation. It is not obvious which scheme works better. If quadratic patches are better than linear, are cubic patches better than quadratic?

Future work will also examine the implementation of surface quadtrees on concurrent machines. Such an implementation could lead to the development of an efficient, interactive system for the design of complex three-dimensional components. An interactive system could also serve as an interactive rendering system for the simulation of physical phenomena, such as mechanics, stress-strain relationships in solids, and molecular modeling.

Hierarchical descriptions of surface shading would permit a wide variety of materials to be modeled ([COOK84]). Surface textures may be mapped onto a surface using a direct projection technique called decal mapping ([BARR83]). A new type of subdivision, color subdivision, may be used to represent the high spatial frequencies present in some texture maps. The decomposition of a surface texture into a quadtree may serve as an attractive initial sampling for a parametric surface. In addition, shadows of surfaces can be rendered using an extension of the boundary subdivision process, using limited ray-tracing to the light source.

It would be useful to model with profile surfaces. These surfaces are formed as the profile product of two parametric curves. ([CARLSON82], [BARR81]). The set of functions may be broadened in the future to include other types of profiles solids, including interpolated profile solids. ([CARLSON82]). Interactive specification of these curves would be attractive. There may be attractive parametric representations for polyhedra; and the set of spherical harmonic functions may be interesting primitives.

Research is still required on how to join intersecting surfaces together, so that cracks do not form along intersection boundaries. Creating fillets at these boundaries may be an attractive approach. It may be possible to smooth inside-outside functions to varying degrees, producing the effect of a low-pass filter. Filleting the intersection boundaries of surfaces would add to the realism achievable with boolean set operations. An exponential weighting of summed implicit algebraic functions can control the sharpness of surface intersections ([BLINN82]). Such smoothing functions may be discovered for other types of primitives.

Boundary subdivision criteria may be improved by studying the conditions under which line segments appear as a smooth curve. This would be useful in forming better termination conditions for silhouette and intersection subdivision.

Appendix A: Surface Quadtree Complexity Theory

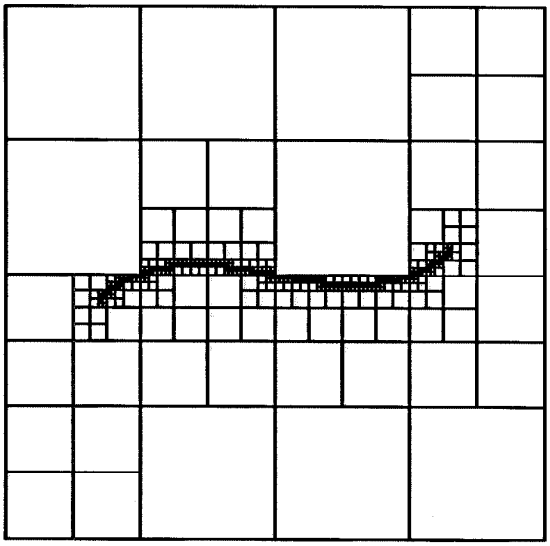


Figure A.1: Unrestricted quadtree approximating a curve.

This appendix proves that the complexity of a restricted quadtree approximating a curve is linear in the curve length and the linear resolution of the quadtree. First, we prove the same result for regular quadtrees, consistent with Samet's result that the quadtree complexity of a one-bit image is linear in the perimeter of the image ([SAMET81]). The proof technique is then extended to the new class of restricted quadtrees.

First we must prove a lemma about the area of a region surrounding a curve.

Lemma 1: A region I of radius R around a planar curve C of length L has an area

$$A \leq \pi R^2 + 2RL.$$

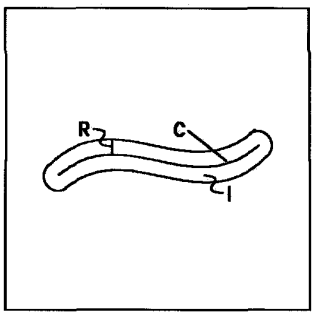


Figure A.2: The area of a region around a curve.

Proof: Subdivide C into n equal parts to give an n-sided open polygon. Each segment has length $ds = L/n$. A circle of radius R is constructed at the beginning of the first segment (Figure A.3). This circle has area πR^2. Another circle of radius R is constructed at the beginning of the second segment, and a rectangle is created joining the two circles. The union of the rectangle and the two circles has area $\pi R^2 + 2R\,ds$. Each time another segment is surrounded, an additional $2R\,ds$ is incorporated into the region. The process is repeated until all the segments have been surrounded. Note that the union of all these regions creates a region surrounding the open polygon by a radius of R.

$L_n \equiv$ length of the n-sided open polygon.

$A_n \equiv$ area of surrounding region $\leq \pi R^2 + n(2R\,ds)$.

Substituting $L = n\,ds$,

$$A_n \leq \pi R^2 + 2RL_n, \quad \text{for all } n.$$

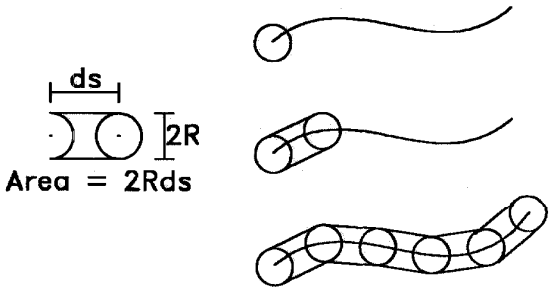


Figure A.3: Construction of a region surrounding curve C.

In the limit as n approaches infinity, we obtain a bound on the area surrounding curve C.

$$\lim_{n\to\infty} A_n \equiv A \leq \pi R^2 + 2RL.$$

If the radius of curvature of C is everywhere greater than or equal to R, and the curve never gets closer than $2R$ to itself, then the equality is exact. Otherwise, the total area is bounded by A.

Theorem 1: Given a curve of length L, the total number of squares N_Q in a quadtree of the curve is

$$N_Q = O\left(\frac{L}{w_{min}}\right),$$

where w_{min} is the smallest square width.

Proof: Let

$g \equiv$ quadtree square generation; $g_{child} = g_{parent} + 1, \quad g_{root} = 0.$

$w(g) \equiv$ quadtree square width; $w(g) = 2^{-g}, \quad g = 0, 1, 2, \ldots.$ Note that $w(g_{root}) = 1.$

$w_{min} \equiv 2^{-g_{max}}, g_{max} \equiv$ maximum generation.

$N_g \equiv$ number of squares of size $w(g)$.

$n_g \equiv$ squares of size $w(g)$ intersecting the plane curve.

Quadtree squares only subdivide when they intersect the curve. They subdivide into four smaller squares, implying $N_{g+1} = 4n_g$.

$L \equiv$ length of the curve approximated by the quadtree.

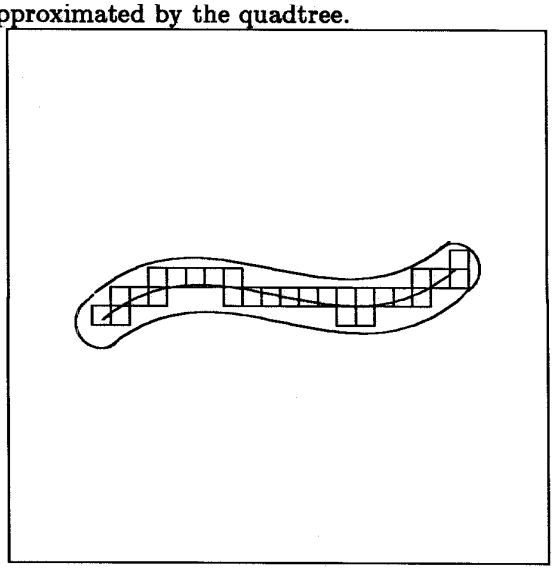


Figure A.4: Intersection region I circumscribing all squares of width $w(g)$ intersecting the curve.

Define an intersection region $I(g)$ which circumscribes all squares of width $w(g)$ intersecting the curve, for some $g \geq 0$. Region I has a radius $R = \sqrt{2}w$ from the curve. Lemma 1 shows that the area of such a region is bounded by

$$A \leq \pi R^2 + 2RL.$$

Substituting $R = \sqrt{2}w$,

$$A_I \leq 2\pi w^2 + 2^{3/2} Lw.$$

The area of a square is $A_s = w^2$. None of the squares of generation g overlap. The region area, A_I, divided by the area of a square of generation g is an upper bound to n_g.

$$n_g \leq \frac{A_I}{A_s} = 2\pi + \frac{2^{3/2} L}{w(g)}$$

The number of squares of size $w(g)$ is given by

$$N_g = 4n_{g-1} \leq 8\pi + 2^{g+5/2}L$$

Summing the number of squares of all sizes,

$$\begin{aligned} N_Q = \sum_{i=0}^{g_{max}} N_i &\leq 8\pi g_{max} + 2^{5/2}L \sum_{i=0}^{g_{max}} 2^i, \\ &\leq 8\pi g_{max} + 2^{5/2}L(2^{g_{max}+1} - 1), \\ &\leq 2^{g_{max}+7/2}L + 8\pi g_{max} - 2^{5/2}L, \\ &= O(2^{g_{max}}L). \end{aligned}$$

Substituting $g_{max} = -\log_2 w_{min}$,

$$N_Q = O\left(\frac{L}{w_{min}}\right).$$

An asymptotic upper bound to N_Q as $g_{max} \to \infty$ is given by $N_Q \leq 8\sqrt{2}L2^{g_{max}}$.

Lemma 2: A restricted quadtree square of width w may be forced to subdivide only if it lies within a manhattan distance of w from a subdivision point.

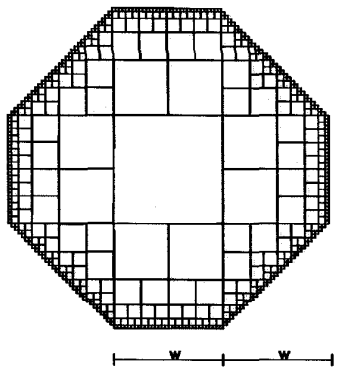


Figure A.5: Geometric progression of a restricted quadtree.

Proof: Restricted quadtree neighbors may differ in size by at most a factor of two. For a square to be forced to subdivide, one of its neighbors must be four times as small. The neighbor's neighbor must be eight times as small, and so on in geometric progression. If the neighbors are positioned end-to-end, the infinite binary progression becomes

$$\frac{w}{2} + \frac{w}{4} + \frac{w}{8} + \frac{w}{16} + \ldots = w.$$

If the subdivision source is greater than w away from the edge of the square, it cannot force the square's neighbor to be smaller than $w/2$. So the square cannot be forced to subdivide.

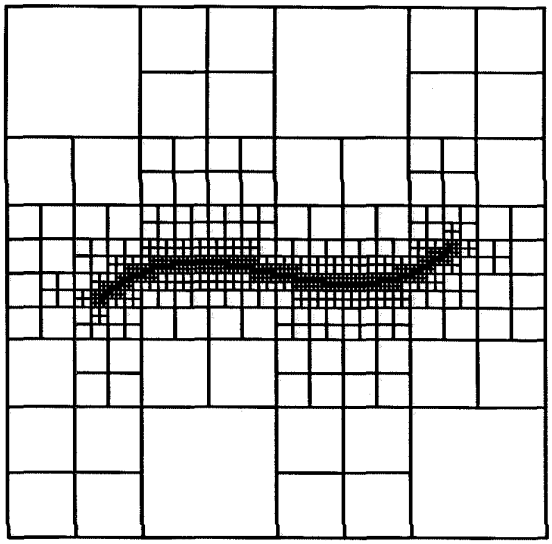


Figure A.6: Restricted quadtree approximating the same curve as in Figure A.1.

Theorem 2: Given a curve of length L, the total number of squares N_{RQ} in a restricted quadtree of the curve is

$$N_{RQ} = O\left(\frac{L}{w_{min}}\right),$$

where w_{min} is the smallest square width.

Proof: The proof is similar to Theorem 1, except that n_g now becomes the number of squares within the radius of influence of the curve. The radius of influence for restricted quadtrees is twice the radius for regular quadtrees, by Lemma 2. Outside the radius of influence, a restricted quadtree square cannot be forced to subdivide.

The radius of the region of influence is doubled for restricted quadtrees, $R = 2R_{unrestricted} = 2\sqrt{2}w$. This at most doubles the number of squares in the quadtree, and does not affect the complexity of the quadtree.

$$N_{RQ} = O\left(\frac{L}{w_{min}}\right).$$

An asymptotic limit to N_{RQ} as $g_{max} \rightarrow \infty$ is given by $N_{RQ} = 16\sqrt{2}L2^{g_{max}}$. This is twice the asymptotic limit for regular quadtrees.

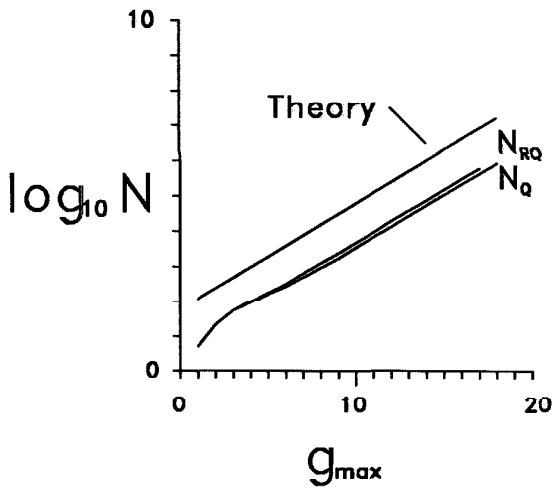


Figure A.7: The measured number of squares in a restricted quadtree N_{RQ} and a regular quadtree N_{RQ} vs. maximum quadtree generation, for the cubic curve in Figs A.1 and A.6. T_{RQ} is the upper bound proven in Theorem 2.

Experimental Results: Figure A.7 shows the number of squares for regular and restricted quadtrees as a function of g_{max}. The quadtrees are of the curves in Figures A.1 and A.6. Both of these functions remain within the expected bounds for a wide range of the maximum quadtree generation g_{max}. The theoretical function T_{RQ} for restricted quadtrees is $T_{RQ} = 16\sqrt{2}L2^{g_{max}}$, corresponding to the most significant term of the bounds for restricted quadtrees. Experimentally, the curve in Figures A.1 and A.6 has a length of $L = 0.68$, where the largest square has unit width. N_Q is approximated by $N_Q = 4.8L2^{g_{max}}$, and N_{RQ} is approximately $N_{RQ} = 6.8L2^{g_{max}}$. The upper limit for regular quadtrees is roughly 2.5 times higher than that observed by the cubic curve in this example. A factor of $\sqrt{2}$ is attributed to the curve being mostly horizontal instead of 45 degrees to the axes. Also, in the average case, the curve goes through one square of width w per curve length w, while in the worse case the same length of curve could pass through two squares of size w. This reduces the average estimate by another factor of 2. Similar effects occur with restricted quadtrees in the average case.

For this example, restricted quadtrees are 50% more expensive than regular quadtrees, in terms of the total number of squares generated, giving some experimental indication of the overhead associated with restricted quadtrees.

Appendix B: Glossary

backface culling: The process of removing parts of a surface that face away from the viewer. The rendering system overhead is reduced by eliminating backfacing polygons before they are drawn.

bicubic patch: A parametric surface defined by cubic equations of two parameters, u and v. The curves generated by varying only one variable u or v are always cubic. The general form of the equation is

$$\begin{aligned}\underline{P}(u,v) = &\underline{A_{11}}u^3v^3 + \underline{A_{12}}u^3v^2 + \underline{A_{13}}u^3v + \underline{A_{14}}u^3 \\ &+ \underline{A_{21}}u^2v^3 + \underline{A_{22}}u^2v^2 + \underline{A_{23}}u^2v + \underline{A_{24}}u^2 \\ &+ \underline{A_{31}}uv^3 + \underline{A_{32}}uv^2 + \underline{A_{33}}uv + \underline{A_{34}}u \\ &+ \underline{A_{41}}v^3 + \underline{A_{42}}v^2 + \underline{A_{43}}v + \underline{A_{44}}\end{aligned}$$

boolean boundary: An intersection curve of two surface manifolds; an intersection boundary.

bivariate function: A function of two variables, such as a parametric surface.

boolean subdivision criterion: A criterion that controls the subdivision of squares on intersection boundaries. For instance, subdivision occurs where the intersection boundary of two surfaces is curved in screen space.

chopping (triangles): Triangles are clipped by the inside-outside function of intersecting parametric surfaces. Triangle chopping reduces a fringe that may occur along the intersection boundary of two surfaces (See Figure 7.A).

culling: The process of removing surface elements from consideration that are not visible for some reason. Common culling steps are backface culling, which removes polygons facing away from the viewer, and surface intersection culling, which removes polygons clipped by other surfaces.

curvature subdivision criterion: A subdivision criterion based on the local curvature of the parametric surface.

deformation: A modeling operation that bends, twists, tapers, or otherwise alters the shape of a parametric surface.

disjoint: Not connected. Two surfaces are disjoint if they do not intersect. Two solids are disjoint if and only if there is no point within the first solid that is also within the second solid.

fixed-point representation: A representation for numbers that has a fixed amount of precision for the integer part and for the fractional part of a number.

hashed array: An array whose entries have been evenly redistributed by a hashing function. This operation is a useful technique in searching problems, where an element in a large array must be found. For example, the samples in a surface quadtree are not evenly distributed in u and v. Hashing on the u and v parameters redistributes the samples so that they evenly populate an array, facilitating storage and accessing operations.

hashing function: A function that maps a record of information onto an array index in a pseu-

dorandom fashion, in an effort to evenly redistribute the elements across the array.

inside-outside function: A function indicating whether a point is inside or outside of a solid. The function returns a negative value if the coordinate is inside the solid, a positive value if the point is outside the solid, and a value of zero if the point is on the surface of the solid. Root-finding techniques converge faster if the inside-outside function is roughly linear with distance to the surface.

intersection boundary: The intersection curve of two surface manifolds.

mach bands: Bright or dark lines caused by artifacts of the human visual system when viewing a continuous intensity function whose slope is discontinuous. Mach bands may be produced at the edges of adjacent smoothly shaded polygons.

manhattan distance: A distance metric defined in two dimensions by

$$d(\mathbf{p}_1, \mathbf{p}_2) = |\mathbf{p}_{2x} - \mathbf{p}_{1x}| + |\mathbf{p}_{2y} - \mathbf{p}_{1y}|,$$

where $\mathbf{p}_1$ and $\mathbf{p}_2$ are position vectors.

modeling hierarchy: A tree of modeling operations such as rotations, translations, scalings, bends, twists, tapers, intersections, and primitives such as quadric surfaces, bicubic splines, half-planes, and profile solids. The operations are combined in a hierarchy specifying the scope of each operation, creating a model of a three dimensional solid.

modeling space: A three-dimensional coordinate system centered around the model being rendered. The curvature subdivision criterion measures curvature in modeling space to control the sampling process.

parameter space: A two dimensional coordinate system spanning the domain of a parametric surface. The coordinates (u, v) are usually used to represent a point in parameter space. The domain of u and v is usually the interval $[0, 1]$.

parameter space wrap-around: Some closed surfaces, such as superquadrics and supertori, are periodic in u and/or v. Cracks may appear in the surface between the edges $u = 0$ and $u = 1$, if care is not taken to match the sample points along each edge.

parametric surface: A vector function $\underline{P}(u, v)$ of two scalar variables that describes a surface embedded in three-dimensional space, or a surface manifold. The parametric functions used in this paper accept values of u and v ranging from 0 to 1.

pixel: A picture element; the smallest unit of a digital image.

planar subdivision: A collection of line segments in the plane, whose pairwise intersections are restricted to segment endpoints. No segment may cross another segment. Planar subdivisions are useful for eliminating cracks in polygonal approximations to parametric surfaces.

principal directions (of curvature of a surface): A pair of local coordinate axes tangent to a parametric surface that point in the directions of maximum and minimum curvature. These axes always lie perpendicular to each other.

profile solid: A solid formed from two curves: a vertical (or north-south) profile, and a horizontal (or east-west) profile.

profile surface: The surface of a profile solid.

proximity subdivision criterion: A criterion that forces subdivision where surfaces potentially intersect. The criterion explores local minima of the absolute value of the inside- outside function.

quadric surface: A surface defined by an algebraic, second degree equation. The quadric surfaces

are the ellipsoids, hyperboloids and paraboloids.

quadtree: A recursively subdivided square, consisting solely of smaller squares.

recursive subdivision: The process of repeatedly subdividing a region until a set of subdivision criteria have been satisfied.

regula falsi **iteration**: An iterative procedure for finding a root of a function, given a positive value and a negative value for the function. The root of the function is estimated to be where the line through the positive and negative sample hits the axis. If this new sample is positive, it replaces the old positive sample; and if it is negative, then it replaces the negative sample. Iteration continues until the value obtained is sufficiently close to zero.

restricted quadtree: A special quadtree, whose neighboring squares may differ in size by at most a factor of two.

root (of a quadtree): The first element of a quadtree, consisting of a single square to be recursively subdivided into smaller squares.

scan-line: A horizontal row of pixels in a digital image that extend from the left border of the image to the right border. Many rendering algorithms compute one scan-line at a time.

screen space: A coordinate system centered on the image being generated by the rendering system. Several subdivision criteria use screen space measurements to determine the visible "badness" of a rendering artifact.

silhouette: The border of a surface as seen from a particular viewpoint.

silhouette subdivision criterion: A criterion governing subdivision of regions that cross the silhouette boundary of a surface.

square (of a quadtree): An element of a quadtree. Every element in the quadtree hierarchy is a square.

superquadric: A quadric surface, whose trigonometric components have been raised to arbitrary exponents.

surface intersection biasing: A technique of offsetting the inside-outside function of a surface so that the polygons of the two surfaces do not touch. This procedure reduces artifacts associated with finite precision arithmetic.

surface manifold: A continuous, piecewise differentiable surface embedded in three-dimensional space.

surface net: A network of parametric surface samples arranged in a simple hierarchy.

surface quadtree: A quadtree that spans the parameter space of a surface. Additional samples are created by subdividing squares into smaller squares.

uniform subdivision criterion: A criterion that forces parametrically uniform sampling of a surface.

unit square: A square defined by the interval $[0, 1]$ in two coordinate directions. The domain of the parametric surfaces used in this paper is bounded by the unit square in parameter space.

References

Barr, A.H., "Superquadric and Angle Preserving Transformations," *IEEE Computer Graphics and Applications*, Volume 1 Number 1, 1981, pp. 41–47.

Barr, A.H., "Geometric Modeling and Fluid Dynamic Analysis of Swimming Spermatozoa", Ph.D. Thesis, Rensselaer Polytechnic Institute, 1983.

Barr, A.H., "Local and Global Deformations of Solid Primitives," *Computer Graphics*, Volume 18, Number 3, SIGGRAPH '84 Proceedings, pp. 21–30.

Barr, A.H., "Inverse Methods in Ray Tracing," in progress.

Blinn, J.F., "Computer Display of Curved Surfaces," Ph.D. Thesis, University of Utah, 1978.

Blinn, J.F., "A Generalization of Algebraic Surface Drawing," *ACM Transactions on Graphics*, Volume 1, Number 3, 1982, pp. 235–256.

Carlson, W.E., "An algorithm and data structure for 3D Object Synthesis using Surface Patch Intersections," *Computer Graphics*, Volume 16, Number 3, SIGGRAPH '82 Proceedings, pp. 255–263.

Catmull, E., "Computer Display of Curved Surfaces," *IEEE Conference Proceedings on Computer Graphics, Pattern Recognition and Data Structures*, May 1975, pp. 11–17.

Cook, R.L., "Shade Trees", *Computer Graphics*, Volume 18, Number 3, '84 SIGGRAPH Proceedings, pp. 223–231.

Hunter, G. M., Steiglitz, K. "Linear Transformation of pictures represented by quadtrees," *Computer Graphics and Image Processing*, Volume 10, 1979 pp. 289–296.

Kirkpatrick, D., "Optimal Search in Planar Subdivisions," *SIAM J. Comput.*, Volume 12, Number 1, February 1983, pp. 28–35.

Lane, J., Carpenter L., "A Generalized Scan Line Algorithm for the Computer Display of Parametrically Defined Surfaces," *Computer Graphics and Image Processing*, Volume 11, 1979, pp. 290–297.

Lee, D.T., Preperata, F.P., "Computational Geometry– A Survey", *IEEE Transactions on Computers*, Volume C-33, Number 12, December 1984, pp. 1086–1087.

Ratliff, F., *Mach Bands: Quantitative studies on neural networks in the retina*, Holden-Day, Inc., 1965.

Samet, H. "Computing perimeters of images represented by quadtrees," *IEEE Pattern Analysis and Machine Intelligence*, Volume 3, Number 6, November 1981, pp. 683–687.

Samet, H. "Neighbor Finding Techniques for Images Represented by Quadtrees," *Computer Graphics and Image Processing*, Volume 18, 1982, pp. 37–57.

Schweitzer, D., Cobb, E.S., "Scanline Rendering of Parametric Surfaces," *Computer Graphics*, Volume 16, Number 3, SIGGRAPH '82 Proceedings, pp. 265–271.

Sederberg, T.W., Anderson, D.C., "Ray Tracing of Steiner Patches," *Computer Graphics*, Volume 18, Number 3, '84 SIGGRAPH Proceedings, pp. 159–164.

Tamminen, M., "Encoding Pixel Trees," *Computer Vision, Graphics, and Image Processing*, Volume 28, 1984, pp 44–57.

Warnock, J.E., "A Hidden-Line Algorithm for Halftone Picture Representation," Ph.D. thesis, University of Utah, 1969.

Watkins, G., "A Real Time Visible Surface Algorithm," Ph.D. Thesis, University of Utah 1970.

Whitted, J.T., "An Improved Illumination Model for Shaded Display," *Communications ACM*, Volume 23, Number 6, June 1980, pp. 343–349.